not. Moniker objects assist in persisting distributed objects. In particular, such distributed objects are typically persisted in a database that is often remote to the client that is using the object. However, in some cases, it is desirable to have a copy of the distributed object persisted in a local storage so that the local state of the client program can be preserved. In accordance with the principles of the invention, when an object is to be released and persisted, a release command is sent to an object model adapter that is part of the distributed object system. In response, the object model adapter creates a moniker object that corresponds to the actual object. This moniker object is then automatically substituted for the actual object by the stream writer when the client streams the real object into the local storage. The moniker object in the local storage can then be maintained by the inventive life cycle services system. This operation is described in detail in the present specification at page 22, line 3 to page 23, line 3 (in connection with figures 13 and 14.)

Similarly, when a distributed object is resurrected in response to a request for the object from the client, the object model adapter causes the object to be created in the remote server. The client also requests the object from the local storage. When the stream reader reads the moniker object from local storage, it automatically substitutes the real object and creates the real object in the client's memory. This operation is described in detail in the present specification at page 23, lines 4-29 (in connection with figures 15 and 16.)

Neither the Atkinson reference nor the Koppolu reference discloses such a substitution operation. Atkinson discloses a system for displaying compound documents that contain data with various data types. In order for a display program, such as a word processing program, to link to the various data types, each data portion with a specific type is represented by a moniker object. The moniker object contains a link to the data and methods for displaying the data. The moniker object also contains a standard interface with which the word processing program can interact. In use, one or more moniker objects are persistently stored with the compound document containing the data represented by the moniker objects. When a word processing program opens the compound document and attempts to display that data, the word processing program first instantiates the corresponding moniker object in the memory and then

uses methods of the instantiated moniker object to load the data to be displayed and to bind to that data (Atkinson, column 14, lines 15-24.) Therefore, it is the moniker object itself that is being stored and resurrected under control of the client (the word processing program). Once the moniker object is in memory, it is then used to load and manipulate the actual data. There is no substitution of the moniker object for another object.

The Koppolu reference teaches a similar mechanism. As taught by Koppolu, a moniker object is used to access another object, such as an HTML or word processing data file. In particular, as disclosed in Koppolu, column 16, lines 17-58, in order to use a moniker object, a client first creates the moniker object in memory either by using an API or by resurrecting the object from storage. Then, the client uses the moniker object methods to bind to the data, which, in turn, launches the object's server application and causes the data object to be instantiated in the server. In this process, the moniker object acquires a pointer to the newly instantiated object. The client can then use the pointer to manipulate the object directly. As with the Atkinson reference, in the Koppolu reference, it is the moniker object itself that is being stored and resurrected under control of the client (generally a browser). Once the moniker object is in memory, it is then used to load and manipulate the actual data. There is no substitution of the moniker object for another object.

Therefore, since neither of the cited references teaches the substitution of the moniker object for another object, the combination of these references cannot teach such an operation. Further, since both references operate in substantially the same manner, the combination cannot suggest the claimed manner of operation.

The present claims explicitly recite that the monitor object is substituted for the distributed object. For example, claim 1, in lines 6-8, recites "a first stream object which automatically substitutes the moniker object for the distributed object when the distributed object is streamed out from the memory to the local storage." Neither of the cited references, nor their combination, teaches or suggests such a stream writer. The examiner points to Koppolu, column 14, line 64 to column 15, line 14 as disclosing a stream object that substitutes a moniker object for a distributed object when the distributed object is streamed from memory to local storage. However, in the section to

which the examiner refers, Koppolu discloses three operations. First, in column 14, lines 55-63, Koppolu states that the moniker object contains stream functions that allow the object to be streamed into and out of storage. Thus, the client can load and store the moniker object. However, Koppolu does not state that during this process, the moniker object is substituted for another object.

Further, in column 15, lines 1-4, Koppolu describes the moniker object's BindToObject function that, as stated, instantiates a named object in the memory. Thereafter the client can use the instantiated object. Therefore, to the extent that the named object is thereby "substituted" for the moniker object in this operation, this substitution occurs when the named object is streamed in from the storage into the memory. Claim 1 recites that the substitution takes place "when the distributed object is streamed out from the memory to the local storage."

Finally, in column 15, lines 4-8 Koppolu describes the moniker object's BindToStorage function that, as stated, instantiates a named object into an OLE storage stream so that the named object is thereby stored in the storage. Note that, in Koppolu, the named object is streamed into the storage whereas as recited in claim 1, the moniker object is substituted for the distributed object during the stream out operation so that the moniker object is stored in the local storage instead of the distributed object. Consequently, claim 1 patentably distinguishes over the cited references.

Claims 11 and 21 contain parallel limitation to those noted above in claim 1 and patentably distinguish over the cited references in the same manner as claim 1.

Claims 2-10 are dependent, either directly or indirectly, on claim 1 and incorporate the limitations thereof. Therefore, they also distinguish over the cited references in the same manner as claim 1. In addition, these claims also recite additional limitations not disclosed or suggested by Atkinson and Koppolu. For example, claim 2 recites that the moniker object is substituted for the distributed object when the distributed object is persisted. As stated above, the Koppolu moniker object persists the named object not itself. Claim 3 recites a second stream object that automatically substitutes a reference to the distributed object for the moniker object when the moniker object is streamed in from local storage. To the extent that Koppolu describes a substitution, this substitution does not occur when the moniker object is

streamed into memory, but when the client calls the moniker object's BindToObject function. Therefore, claims 2 and 3 patentably distinguish over the cited references.

Claims 12-20 are dependent, either directly or indirectly, on claim 11 and incorporate the limitations thereof. Therefore, they also distinguish over the cited references in the same manner as claim 11. In addition, these claims also recite limitations that parallel the limitations in claims 2-10 and thereby distinguish over the cited references in the same manner as claims 2-10.

Finally, claims 22-30 are dependent, either directly or indirectly, on claim 21 and incorporate the limitations thereof. Therefore, they also distinguish over the cited references in the same manner as claim 21. In addition, these claims also recite limitations that parallel the limitations in claims 2-10 and thereby distinguish over the cited references in the same manner as claims 2-10.

Applicant believes the claims are now in allowable condition. A notice of allowance for this application is solicited earnestly. If the examiner has any further questions regarding this amendment, he/she is invited to call applicant's attorney at the number listed below. The examiner is hereby authorized to charge any fees or credit any balances under 37 CFR §§1.17, and 1.16 to Deposit Account No. 09-0460.

Respectfully submitted

Paul E. Kudirka Date: 5/8/03

Paul E. Kudirka, Esq. Reg. No. 26,931
KUDIRKA & JOBSE, LLP
Customer Number 021127
Tel: (617) 367-4600 Fax: (617) 367-4656